View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Newcastle University E-Prints

Alssaiari A, Thomas N.

Performance Modelling of Dynamic Server Allocation for Energy Efficiency using PEPA.

In: 32nd UK Performance Engineering Workshop.

8-9 September 2016, University of Bradford.

This is the authors' accepted manuscript of a paper that was presented at the 32nd UK Performance Engineering Workshop.

Conference website:

http://computing.brad.ac.uk/ukpew2016/

Date deposited:

31/10/2016

This work is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported License

Newcastle University ePrints - eprint.ncl.ac.uk

Performance Modelling of Dynamic Server Allocation for Energy Efficiency using PEPA

Ali Alssaiari* Nigel Thomas†

Abstract

Power consumption in data centres receive a huge concern by data centre providers. This project, consider modelling a policy in PEPA to control powering on or off servers dynamically. The main focus was to identify and reflect over the trade-off between saving energy (by powering down servers) and performance cost. While powering down servers save energy, it could increase the performance cost. The experiment analyses the effect of the policy on energy consumption and performance cost. Different combination of dynamic and static servers used in this policy against different scenarios including changes job arrival rate, job arrival duration and the time that is needed by servers to fully power on and serve jobs. This gives an interesting outcome because every scenario is unique and therefore no server combinations does gives low energy and high performance in all scenario.

1 Introduction

The cost of energy is one of the many challenges facing large-scale computing. Data centre owners now expect to spend more capital on energy than their IT infrastructure, which currently contributes more to the total cost of ownership (TOC). The Environmental Protection Agency (EPA) has issued a report to the U.S. Congress about the energy efficiency of servers and data centres. The report highlighted several important points related to the energy consumption of data centres. According to the report, data centres electricity demands grew 100 % between 2000 and 2006. Data centres in the U.S. consumed 61 billion kWh in 2006, representing 1.5% of total electrical consumption in the country [1]. In this paper we focus on the trade-off between performance and power consumption based on previous efforts in simulation and queueing theory [2,3]. If we want to reduce energy use but maintain quality of service then we essentially have two choices, either we manage the workload, e.g. [4,5] or we dynamically manage the servers we have available e.g. [6-10]. In this paper we present a PEPA model of a policy for managing the turning on and off of servers, previously studied in [2] and [3]. Various assumptions and modifications to the original model have been made to help analyse the policy model in PEPA.

*School of Computing, Newcastle University, a.alssaiari@ncl.ac.uk
†School of Computing, Newcastle University, nigel.thomas@ncl.ac.uk

2 Related Work

The problem of unnecessary power consumption in data centre by servers has been the subject of increasing focus; however, the amount of energy that can be saved by switching servers on or off according to demand has not been given more consideration. Slegers et al [3] propose a number of heuristics in this regard that organise server usage according to demand. This approach is further examined in [2], by using JAVA to implement the heuristics polices and simulate servers allocation power efficiency management according to demand. The heuristics referred to in [2] were simulated experimentally. The heuristics were individually tested by running through the circumstances established in the design of the experiments. A variety of experimental scenarios were established in order that the heuristics could be adequately appraised; this was of particular importance as the conclusion presented in [2] determined that no heuristic is universally applicable, but that each heuristic qualities that are appropriate in only certain circumstances.

The system in [2] consists of a given number (N) of homogeneous servers. These servers may be in any of five conditions; namely, powered up, wherein the server may be either active or idle, powering up, powered down, powering down and fault. While in the powered down condition the server may be in a quiescent condition or completely turned off although, whichever happens to be the case, its power consumption would be little or nothing. In the transitional conditions, i.e. powering down or powering up, and the fault condition, servers are unable to respond to service demands but would nevertheless still be consuming power. Although the fault condition could occur simultaneously with any of the other four conditions, the author of [2] only considered the fault in the transitional states (Powering up and Powering down) due to the increased likelihood of faults becoming apparent then. This, it is surmised, should be sufficient in achieving an indication of how consistently the system is performing. In order to analyse the relative costs inherent in system operation, costs were assigned to each system condition. These were as follows..Powered up: C_{up} , Powering up: C_{pow}Up ,Powered down: C $_{idle}$ and C_{down} ,Powering down: C_{pow}Down and Fault: C_{fault} .The first four of these represent relative power costs per unit time, while C_{fault} represents the proportionate detriment suffered by the system as a result of a server being faulty and, consequently, inoperative.

An additional cost was identified in respect of system operation, this being C_{job} the cost of holding jobs in the queue for in excess of one unit time. This represented a requirement of increasing the rate at which jobs were handled in order to meet increased service demand.The final additional cost identified was C_{pow} the beneficial consequence of holding a server in the powered down condition per unit time. These costs are expressed comparatively so, if C_{pow} =1 and C_{job} = 2, it may be surmised that the cost associated with holding a job in the queue is twice that of the benefit accrued as a result of holding a server in the powered down condition.

There are a number of heuristics noted in [2] and [3], each of which has its own advantages and disadvantages with regard to power conservation, optimal performance and consistent performance. In total there are a six policies introduced in [3] and simulated in JAVA in [2]. In this paper only one heuristic was considered out of the six heuristics to be discussed and modelled using PEPA.

2.1 High/Low Heuristic

The high/low heuristic bears comparison with the semi-static allocation heuristic, but in this instance transition periods are included in the heuristics determinations. The high/low heuristic also builds arrival periods, job accomplishment times and job queue length into its decision-making processes, and in consequence is the most complex. Despite its inherent complexity, the high/low heuristic maintains its stability as a result of job arrival rates still being categorised in high or λ_{low} binary form and the job time μ being maintained as a constant. The high/low heuristic is a complex application that incorporates system performance and stability into its determinations.

3 Performance Evaluation Process Algebra

Performance Evalauation Process Algebra (PEPA) is an extension to classical process algebras developed to be a high level description language for Markov processes [11]. In PEPA, systems are represented as set of components that can be either individually or multiply engaged in activities. Each component represents an element of the system or behaviour. The occurrence of each activity in the system is determined by the associated rate for each action in that activity. An activity (*arrival*, λ) consists of the action type arrival associated with the activity rate λ. The supported rate of any action by PEPA follows the negative exponential distribution. Thus, the rate λ can be any positive real number or unspecified (represented as T). Unspecified, or passive, rate means the component does not have control over the rate of this action. For expediency, a set of combinatory symbols are used to construct the behaviour of the system. Consider the following simple example:

$$\begin{aligned} P_1 &\stackrel{def}{=} (request, \lambda).P_2 + (think, \alpha).P_1 \\ P_2 &\stackrel{def}{=} (service, \mu).P_1 \end{aligned}$$

In this example the component *P1* either engages in an activity (request,) to perform an action type request at rate and then behaves as *P2*, or it engages in an activity (*think*,α) and remains *P1*. *P2* performs activity (service.μ) with action type *service* at rate μ and returns to *P1*. Single components such as this are not particularly useful in isolation, but can be combined to form a model using the synchronisation operator $\bowtie$ over a cooperation set. Consider the cooperation

$$P \underset{\mathcal{L}}{\bowtie} Q$$

Here two components P and Q cooperated over a set of actions L. This means that P and Q can only engage in any action type in L if both of them engage in the action at the same time. Such an action is referred to as a shared action. The rate of shared actions is determined by the minimum of the rates for the action specified in P and Q. For the purposes of the rate calculation an unspecified action rate is interpreted as $\top$. Only shared actions may have an unspecified rate and at most one component may have an unspecified rate in any cooperation. Hence the resultant rate for any action, shared or local, must be finite. A formal specification of PEPA can be found in [11]. The PEPA model used in this paper is essentially a multi-server queue. A discussion of approached to modelling queues in PEPA is given in [12].

4 The Modle in PEPA

In this paper we consider a variant of the high/low policy presented in [2, 3]. The model is specified in PEPA as follows:

$$
\begin{aligned}
Q_0 &\stackrel{def}{=} (arrivalH, \lambda).Q_1 + (arrivalL, \epsilon).Q_1;\\
Q_i &\stackrel{def}{=} (arrivalH, \lambda).Q_{i+1} + (arrivalL, \epsilon).Q_{i+1}(service, \mu).Q_{i-1}; \quad 0 < i < n\\
Q_n &\stackrel{def}{=} (service, \mu).Q_{n-1};
\end{aligned}
$$

$$
\begin{aligned}
Arrival_{high} &\stackrel{def}{=} (arrivalH, \lambda).Arrival_{high} + (highPeriodEnd, \beta).Arrival_{low};\\
Arrival_{low} &\stackrel{def}{=} (arrivalL, \epsilon).Arrival_{low} + (lowPeriodEnd, \gamma).Arrival_{high};
\end{aligned}
$$

$$
\begin{aligned}
ServerPowering_{on} &\stackrel{def}{=} (powerup, \eta * (1-\rho)).Server_{on} + (powerup, \eta * \rho).Server_{failOn};\\
Server_{on} &\stackrel{def}{=} (service, \mu).Server_{on} + (highPeriodEnd, \beta)ServerPowering_{off};\\
ServerPowering_{off} &\stackrel{def}{=} (poweroff, \xi * (1-\rho)).Server_{off} + (poweroff, \xi * \rho).Server_{failOff};\\
Server_{off} &\stackrel{def}{=} (lowPeriodEnd, \gamma).ServerPowering_{on};
\end{aligned}
$$

$$
\begin{aligned}
Server_{static} &\stackrel{def}{=} (service, \mu).Server_{static};\\
Server_{failOn} &\stackrel{def}{=} (repair, \sigma).Server_{on};\\
Server_{failOff} &\stackrel{def}{=} (repair, \sigma).Server_{off};
\end{aligned}
$$

$$
(Arrival_{high} \underset{\mathcal{K}}{\bowtie} Server_{on} \underset{\mathcal{K}}{\bowtie} ... \underset{\mathcal{K}}{\bowtie} Server_{on}) \underset{\phi}{\bowtie} Server_{static}[m]) \underset{\mathcal{L}}{\bowtie} Q_0
$$

The number of jobs in the system at the current time, i, is specified by the current state of Q_i. The maximum number of jobs is bounded at N. Arrivals into the system occur at either a high rate λ, by action *arrivalH,* or at a low rate ε (typically zero) by action *arrivalL*. The arrival stream switches between the high and low rate through the actions *highPeriodEnd* and l*owPeriodEnd* at rates β and γ respectively. Hence a high period has duration $1/\beta$ and the low period has duration $1/\gamma$. Thus the average arrival rate is given as

$$\overline{\lambda} = \frac{\lambda\gamma + \xi\beta}{\gamma + \beta}$$

Jobs leave the system according to the service process, which is determined by the number of active servers. M servers are static and remain permanently available to serve jobs. The remaining servers turn on and off in response to the high and low periods of arrivals. Thus, when a high period ends these dynamic servers will become unavailable for service, but when a low period ends they will turn back on. It is assumed that there is a delay in turning servers on and off (rates η and ξ respectively). Therefore when a high period begins there will be a delay until the dynamic servers are available to serve jobs. Clearly if this delay is large and the high arrival rate greatly exceeds the service capacity of the static servers then there may be a significant increase in the number of jobs in the system during this time. During the turning on and turning off periods servers will continue to consume power but are not providing a service. Hence if these periods are long then they would represent a potentially significant inefficiency in the system.

It is further assumed that servers may fail when switching on and off with probability p. Following failures, servers undergo repair at rate σ and it is

assumed that during repair the servers will consume energy as if they are working normally. We distinguish whether servers fail in the turning on or turning off state so that following repair the server will return to the same state as other dynamic servers. It is assumed that all dynamic servers must begin to turn on or off simultaneously. Therefore a *highPeriodEnd* action may only occur if all the dynamic servers are on and likewise a *lowPeriodEnd* action may only occur when all the dynamic servers are off. Clearly such a synchronisation is not ideal, however typically the switching rates of the arrival on and off periods are much longer than the switching periods needed to power servers on and off; if they werent powering off would not be a sensible option. Hence this synchronisation between the server state and the arrival state would not affect the average arrival rate unless the failure rate is high and the average repair time is excessive.

The problem associated with this model is to find the optimal number of static and dynamic servers needed to minimise the energy usage for a given set of parameters (arrival rates, service rate, switching rates, failure probability and repair rate). We will explore this problem in the next section.

5 Evaluation

Experiments are conducted based on several scenarios. These scenarios represent various datacentre environments and preferences. The first scenario compares difference combinations of dynamic and static servers against different high arrival rates. The second scenario compares the same combinations of servers applied in the first scenario against different durations of high arrival periods, where the arrival rate is high but the durations are varied. The third scenario varies the switching time for powering the servers on or off. The fourth scenario considered variations in the costs of holding jobs in the queue and the benefits of saving energy by powering down servers while varying different ratios. Finally, the total fault in all server combinations was examined and presented to describe which combination could be the best, in terms of stability.

Two costs are considered in this paper energy cost and performance cost. Performance cost is represented by the queue size. The energy cost is the cost of energy consumed by all servers, except when they are powered down or in a repair state. Building the model in PEPA spared the use of a lot of the calculation involved in obtaining the queue size. The queue size in this paper is limited to 16 queues. The energy cost used below is an updated formula that takes more servers states in its consideration.

- Energy Cost:

$$C1 * (Server_{on} + ServerPowering_{on} + ServerPowering_{off} + Server_{static}$$

- Performance Cost:

$$C2 * (1 - Prob(Q_0))$$

$C1$ represents the weight assigned to energy, while $C2$ is the weight assigned to the performance. These two weights are used to indicate the costs that is used to trade-off the relative merits of performance and energy usage. As such, where a data centre values performance over energy saving, $C2$ will be assigned a higher

value than $C1$. However, to have an indicator for comparison between different combinations of servers, the total of both costs (energy and performance) are used to represent the overall performance cost. Where total costs are low in any server combination, it is said to have the lowest overall performance cost, making it a better preference for data centres.

The experiment was repeated for each combination of servers against each specified value of arrival rates, length of period and switching time. The findings from each combination of servers in each scenario were then compared to analyse their performance and energy saving.

5.1 Increasing Arrival rate

In this scenario, the experiment configuration and settings for the average request processing time is set to 12. The high arrival rate was increased from 10, representing the low arrival rate, to 50, which uses high server capacity (resources) at the highest arrival rate. The low arrival rate was set to 10. In addition, the high arrival time and low arrival time were set to 10, meaning both periods are neither too short nor too long. The holding job cost $C2$ and benefit of servers being down $C1$ is set to 1. In this case, the data centre does not prioritise energy over performance and vice versa. The probability of failure during powering up or down servers was set up at 10%, which is sufficiently high for this experiment environment and system model to monitor the effect of servers failure on the system. Finally, the rate of powering up and down servers were set to 100 for both states, which means switching servers on or turning them off will take a short period of time.

The experiment has been carried out by increasing the high arrival rate in a repetitive fashion on each combination of dynamic and static servers. Figure 1 shows the lowest total cost at the highest arrival rate of 50 is given 3 dynamic servers and 3 static servers. The lowest total cost at the arrival rate of 10 occurs where there are 5 dynamic servers and 1 static server. The reason for the 3 dynamic servers and 3 static servers better performance than the other server combinations in total cost can be attributed to the decline in the performance cost at a high arrival rate with this combination. From Figure 2 it can be observe that performance cost increases when there are more dynamic servers, say five, at a higher arrival rate of 50. The reason for that is the increases in waiting jobs in the queue. This is evident because, in this model, the decision to switch on additional servers to complete the job depends on the duration of the period. Section 5.2 takes an in-depth look at the impact of different durations of high arrival periods on the system. In fact, an investigation shows the 5 dynamic servers perform better in terms of energy saving in this experiment compared to other combinations of dynamic and static servers Figure3.

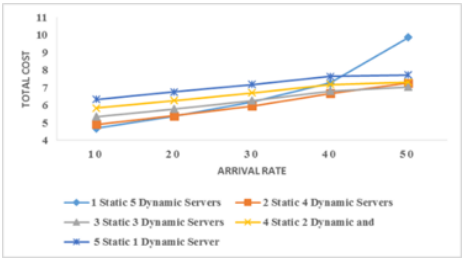

Figure 1: Total Cost Based on Number of Dynamic Servers at Different Arrival rates

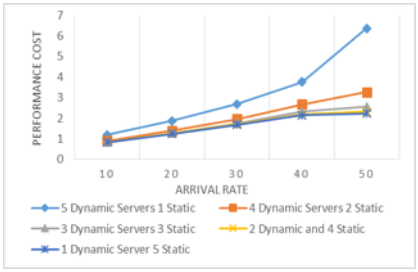

Figure 2: Performance Cost based on Number of Dynamic Servers at Different Arrival Rates

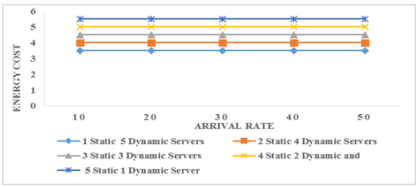

Figure 3: Performance Cost based on Number of Dynamic Servers at Different Arrival Rates

5.2 Changes in Period Duration

In this experiment, the high arrival and low arrival rates were set at fixed values of 50 and 10 respectively. Six (N=6) servers were used in this model, with five turned on or off depending on the duration of the high arrival period, which the last is a static server that is always on to serve jobs in high and low arrival periods. With an average processing time of job requests of 12, the rates for powering up and down servers were set to 100 for both states. The probability of system failure during transition states were 10%, which can be considered acceptable for this small model.

In this scenario, the high arrival period duration rate is varied at 0.1, 1, 10, and 100. These rates represent a longer period at a small rate and a short and faster period at a bigger large rate. The objective of altering the duration is to show the impact of unstable durations of high arrivals. The idea is to analyse the impact of a short high arrival period on system performance. For instance, a high arrival period could be reached before deciding whether to power on additional servers to serve the job. As a result, the waiting jobs in the queue will grow, leading to increased performance costs.The job holding cost $C2$ was considered in this scenario to be 1 and the energy benefits of powering down servers $C1$ as 1. This set up means the data centre does not prioritise energy saving over performance and vice versa.

Figure 4 shows that total cost decreases when the duration is shorter, e.g. 100, while Figure 5 shows that an increase in the job queue leads to a corresponding increase in performance cost.This increase in performance cost has not been reflected in the total cost due to the higher decrease in energy cost.Figure 6 combines the two preceding graphs to show the correlation between the fac-

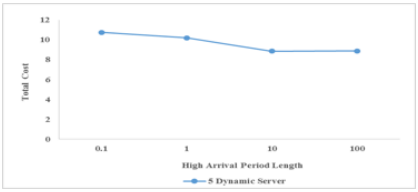

Figure 4: Total cost at different period length

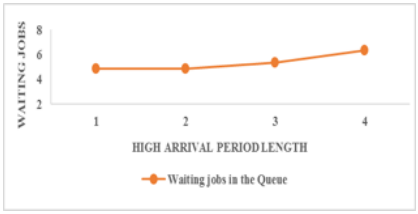

Figure 5: Queue length at different period length rates

tors. Figure 7 shows that when the arrival period is short, the system does not have sufficient time to switch on additional servers compared with long arrival durations. This leads to a decrease in energy cost.

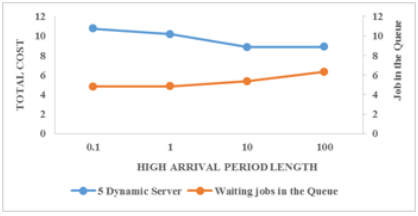

Figure 6: Total cost and queue length for different period rates

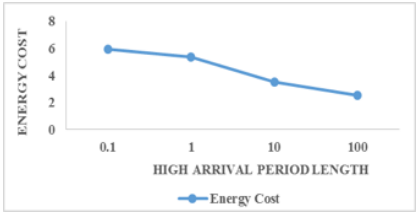

Figure 7: Energy cost based on high arrival period duration

A further analysis conducted by varying the combination of servers that are dynamic and static for each rate of the length of period. The objective is to map out the impact of different durations with different server combinations to highlight the best energy and total cost savings. Figure 8 below indicates that at the longest high arrival period (e.g. 0.1), the energy cost for all server combinations are similar. However, at shorter periods, the energy cost becomes lower when there are more dynamic servers involved. Figure 9 below show the total cost for the same settings, confirming that at the shortest duration, a combination of 5 dynamic and 1 static servers leads to higher total costs. However, this does not mean that having the least number of dynamic servers will reduce total cost. For example, 5 static and 1 dynamic servers have higher costs than 3 static and 3 dynamic servers. This is because the short period ceases before sufficient servers could be switched on to serve the jobs. So, the energy cost for 3 dynamic servers is much less than 5 static servers .Hence, the total cost with 5 static servers is more than the total cost with 3 dynamic servers and 3 static.

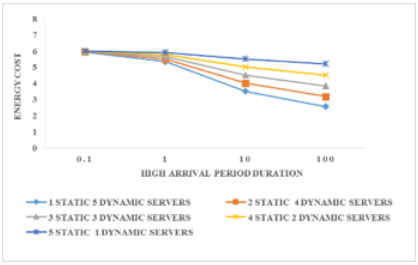

Figure 8: :Energy Cost Based on Number of Dynamic Servers at Different Period length rates

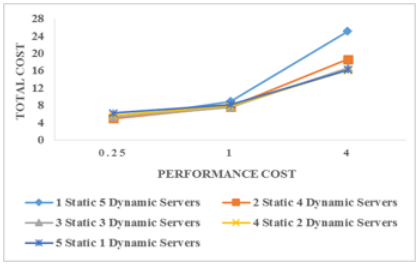

Figure 9: Total Cost based on number of dynamic servers at different period rates

5.3 Changing the switching time

In this case, the experimentation was configured in the same way as in section 5.1. However, instead of varying the high arrival periods, the switching time between having the servers on and off vary at 0.1, 1, 10, and 100. Here, the numbers are ordered from a slower to faster switching time. The high arrival period duration was set to 10, which is neither too short nor too fast. For this configuration, we considered five dynamic and one static server because it is helpful to observe servers switching states (on or off) in relation to switching time. Hence, no other combinations of servers are used. At faster switching times, the overall performance cost is lower. Interestingly, the Figure 10 shows that long switching time leads to an increase in performance cost .This is because servers with slower switching times being unable to respond to more job requests, which causes a backlog of jobs waiting in the queue. Consequently, having more jobs in the queue increases the performance cost, which increases the total cost as a result. On the other hand, the performance cost is lowered with faster switching times as the server can be switched on and respond to job requests quickly, which avoids any delay that causes a queue backlog. The energy cost declines with a fast switching time, as shown in 11 below. Further investigation reveals that a fast switching time reduces the number of servers that are in powering on or off states, helping to lower energy costs.

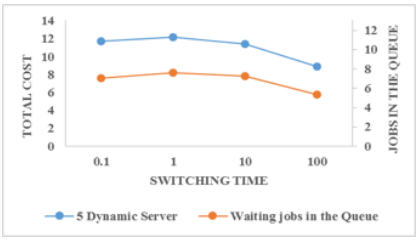

Figure 10: Total cost and waiting jobs for 5 dynamic servers at different switching time Rate

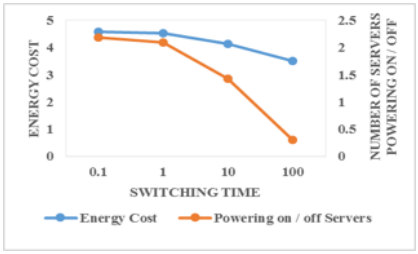

Figure 11: Energy cost and number of servers in powering on / off states at different switching time rates.

5.4 Changing the Cost Deference

In all previous scenarios in this document, the set up does not prioritise energy over processing the jobs in the queue. However, here we look at various data centre options for deciding between energy and performance. The same set up used in section 5.1 is applied in this scenario for the arrival rate and powering on or off state rates. How- ever, the length of period was set to 10. In addition, the benefits of saving energy $C1$ is constant at 1 while the cost of holding job in the queue or performance cost increased from 0.25, 1 and 4.

When the cost of holding job is higher than the cost of saving energy, the total cost is higher with a larger number of dynamic servers. Figure 12 below shows the combination of 5 dynamic servers and 1 static server performs worse than other combinations, and even more poorly than a combination of 5 static servers and 1 dynamic. This is because having more dynamic servers during a period duration that is not too long leads to increased performance cost by having more waiting jobs in the queue, which increases the total cost. Even though there are energy savings when there are more dynamic servers used, Figure 13 below shows the energy saved is not as big as the relative increase in performance cost, thus energy cost becomes less profound in the total cost.

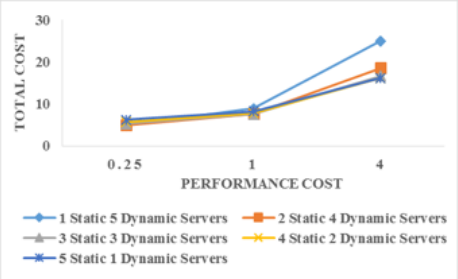

Figure 12: Changing cost difference

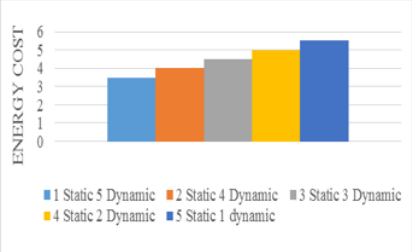

Figure 13: Energy cost for different server combinations

5.5 The fault rate

Fault rates are considered in order to determine data centre performance accurately due to the impact of switching state failures (on or off). In this scenario, the same set up as section 5.2 was used, with a fault rate of 10%.

The fault population increased dramatically when the duration of the high arrival period is short. This can be understood because when the length of period is short, more switching on requests are initiated in a short time. However, before servers are switched on, the period ends and sends a request to switch off servers. Hence, there is an increased risk of server failure at this moment. This can be clearly seen when there are more servers set up to be dynamic servers.

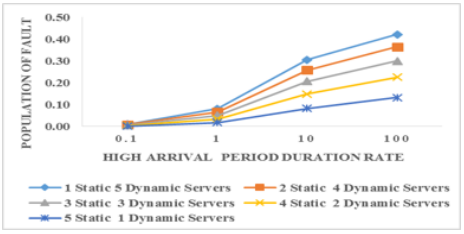

Figure 14: Changing cost difference

5.6 Evaluation Summary

The experiment results from each scenario demonstrate that no one combination per- forms well in all different scenarios. However, each has its own strengths and weaknesses. The combination of 5 static servers and one dynamic is the most stable combination because it does not involve a high level of switching. The combination of 5 dynamic and 1 static server performs well in decreasing energy costs when the period is short, but very poorly in terms of performance as it increases the total cost by in- creasing the waiting job in the queue. However, this combination performs well when the switching time between servers is too fast. The combination of 3 dynamic servers and 3 static servers provides a better overall performance than the other combinations with an increased arrival rate, giving us a mid-position in terms of energy saving.

There is clearly no one combination of servers that suits every condition, as each is adoptable in one scenario but not necessarily good in another. The performance of each combination is affected by many factors. These factors in- clude, the difference between arrival rates, length of period, probability of faults, and switching time. So, the choice of which combination to use for data centres depends on their preference for performance or energy saving. So, the operator of the data centre can configure the combination of servers according to the situation to have a balance between the performance and energy consumption.

6 Conclusions and Future Work

In this paper a policy to limit power consumption by servers in data centres was discussed and modelled in PEPA. Numerical experiments were carried out under different scenarios. In each scenario there was more than one combinations of servers in order to find out which combination is better under each operating conditions. The result of the conducted experiments shows that there is no one combination of servers under this policy that can perform well in all situations.

Our model in PEPA has some limitations. Firstly, the model depends only on the length of arrival period to take the decision of switching on or off more servers. Secondly, we assumed all servers in any state consume the same amount of energy. However, it is not true in practice as servers processing jobs consume more energy than servers in powering on or off state. Moreover, not all servers in data centres are identical in hardware which means each server consume different amount of energy. Finally, the experiment setup in some scenarios changes only one factor and fixed the others, which in practice may not be the situation. One particular scenario worth considering is where all servers are overloaded during

short intense high arrival periods and then there are no arrivals for a very long time. In this situation we would want to save power during the long low periods, but we would not want to switch off all the dynamic servers immediately as there would be a lengthy backlog of jobs to process.

Future work in this direction could look at an extended model to consider more servers with extended queue size. In addition, each server state could be assigned different energy costs to represent as close as possible the actual cost of energy consumption. Finally, it would be useful to change more than one factor at a time instead of changing only one factor with other factors fixed as in some scenarios. Clearly there are many other options for managing servers which remain to be explored using the approach presented in this paper.

References

[1] Brown, R., Report to congress on server and data center energy efficiency: Public law 109-431, Lawrence Berkeley National Laboratory, 2008.

[2] T.H. Nguyen, M. Forshaw and N. Thomas, Operating policies for energy efficient dynamic server allocation, Electronic Notes in Theoretical Computer Science, 318, 159-177, 2015.

[3] J. Slegers, N. Thomas and I. Mitrani, Dynamic server allocation for power and performance, in: Performance Evaluation - Metrics, Models and Benchmarks: SPEC Intl Performance Evaluation Workshop, LNCS 5119, Springer, 2008.

[4] K. Gilly, C. Juiz, N. Thomas and R. Puigjaner, Scalable QoS content-aware load balancing algorithm for a Web Switch based on classical policies, in: 22nd IEEE Intl Conf. on Advanced Information Networking and Applications, IEEE, 2008.

[5] K. Gilly, C. Juiz, N. Thomas, R. Puigjaner, Adaptive Admission Control Algorithm in a QoS-aware Web System, Journal of Information Sciences, 199, 58-77, 2012.

[6] J Slegers, I Mitrani and N Thomas Evaluating the optimal server allocation policy for clusters with on/off sources Performance Evaluation 66(8), 453-467, 2009.

[7] I Mitrani, Managing performance and power consumption in a server farm, Annals of Operations Research 202(1), 121-134, 2013.

[8] T. Phung-Duc, Server farms with batch arrival and staggered setup, in: Proc. of the 5th Symposium on Information and Communication Technology, ACM, 2014

[9] T. Phung-Duc, Single-server systems with power-saving modes, in: Proc. International Conference on Analytical and Stochastic Modeling Techniques and Applications. Springer, 2015

[10] T. Phung-Duc and K. Kawanishi, Energy-Aware Data Centers with s-Staggered Setup and Abandonment, in: Proc. International Conference on Analytical and Stochastic Modeling Techniques and Applications. Springer, 2016

[11] J. Hillston. A compositional approach to performance modelling. Vol. 12. Cambridge University Press, 2005.

[12] N Thomas and J Hillston, Using Markovian process algebra to specify interactions in queueing systems, UKPEW, University of Edinburgh, 1998.